```
EEEEEEEEEEEEEEEE    RRRRRRRRRRRR         FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEEE    RRRRRRRRRRRR         FFFFFFFFFFFFFFFF
EEEEEEEEEEEEEEEE    RRRRRRRRRRRR         FFFFFFFFFFFFFFFF
EEE                 RRR          RRR     FFF
EEE                 RRR          RRR     FFF
EEE                 RRR          RRR     FFF
EEE                 RRR          RRR     FFF
EEE                 RRR          RRR     FFF
EEE                 RRR          RRR     FFF
EEEEEEEEEEEE        RRRRRRRRRRRR         FFFFFFFFFFFF
EEEEEEEEEEEE        RRRRRRRRRRRR         FFFFFFFFFFFF
EEEEEEEEEEEE        RRRRRRRRRRRR         FFFFFFFFFFFF
EEE                 RRR    RRR           FFF
EEE                 RRR    RRR           FFF
EEE                 RRR    RRR           FFF
EEE                 RRR       RRR        FFF
EEE                 RRR       RRR        FFF
EEE                 RRR       RRR        FFF
EEEEEEEEEEEEEEEE    RRR          RRR     FFF
EEEEEEEEEEEEEEEE    RRR          RRR     FFF
EEEEEEEEEEEEEEEE    RRR          RRR     FFF
```

```
TTTTTTTTTT    SSSSSSSS    TTTTTTTTTT    AAAAAA      PPPPPPPP    EEEEEEEEEE
TTTTTTTTTT    SSSSSSSS    TTTTTTTTTT    AAAAAA      PPPPPPPP    EEEEEEEEEE
    TT          SS            TT        AA    AA    PP    PP    EE
    TT          SS            TT        AA    AA    PP    PP    EE
    TT          SS            TT        AA    AA    PP    PP    EE
    TT          SS            TT        AA    AA    PP    PP    EE
    TT        SSSSSS          TT        AA    AA    PPPPPPPP    EEEEEEE
    TT        SSSSSS          TT        AA    AA    PPPPPPPP    EEEEEEE
    TT            SS          TT        AAAAAAAAAA  PP          EE
    TT            SS          TT        AAAAAAAAAA  PP          EE
    TT            SS          TT        AA    AA    PP          EE
    TT            SS          TT        AA    AA    PP          EE
    TT        SSSSSSSS        TT        AA    AA    PP          EEEEEEEEEE      ....
    TT        SSSSSSSS        TT        AA    AA    PP          EEEEEEEEEE      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II            SS
LL                II            SS
LL                II            SS
LL                II            SS
LL                II          SSSSSS
LL                II          SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS
```

```
0001      C
0002      C Version:      'V04-000'
0003      C
0004      C*****************************************************************
0005      C*                                                              *
0006      C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                     *
0007      C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.      *
0008      C*  ALL RIGHTS RESERVED.                                        *
0009      C*                                                              *
0010      C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0011      C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH LICENSE  AND WITH THE   *
0012      C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0013      C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0014      C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0015      C*  TRANSFERRED.                                                *
0016      C*                                                              *
0017      C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0018      C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0019      C*  CORPORATION.                                                *
0020      C*                                                              *
0021      C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0022      C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.     *
0023      C*                                                              *
0024      C*                                                              *
0025      C*****************************************************************
0026      C
0027
0028            SUBROUTINE TSTAPE (LUN)
0029
0030      C
0031      C     AUTHOR  BRIAN PORTER            CREATION DATE   20-SEP-1979
0032      C
0033      C     Functional description:
0034      C
0035      C     TS11 tape handler device error/timeout display module
0036      C
0037      C     Modified by:
0038      C
0039      C     V03-005 SAR0235          Sharon A. Reynolds,    28-Mar-1984
0040      C             Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
0041      C
0042      C     V03-004 SAR0099          Sharon A. Reynolds,    20-Jun-1983
0043      C             Changed the carriage control in the 'format' statements
0044      C             for use with ERF.
0045      C
0046      C     V03-003 SAR0045          Sharon A. Reynolds,    9-Jun-1983
0047      C             Remove brief/cryptic support.
0048      C
0049      C     v03-002 BP0007           Brian Porter,          20-AUG-1982
0050      C             Minor edit.
0051      C
0052      C     v03-001 BP0006           Brian Porter,          05-MAR-1982
0053      C             Corrected tssr and tsba in 'b' and 'c' formats.
0054      C**
0055
0056            INCLUDE 'SRC$:MSGHDR.FOR /NOLIST'
0115            INCLUDE 'SRC$:DEVERR.FOR /NOLIST'
```

```
0216
0217
0218          BYTE            LUN
0219
0220          INTEGER*4       MODE
0221          INTEGER*4       FIELD
0222          INTEGER*4       COMPRESS4
0223          INTEGER*4       COMPRESSC
0224          INTEGER*4       TSBA
0225          INTEGER*4       TSSR
0226          INTEGER*4       TS_FUNCTION
0227          INTEGER*4       UBA_REGS(0:4)
0228          INTEGER*4       CMD_BUF(0:7)
0229          INTEGER*4       MSG_BUF(0:6)
0230          INTEGER*4       MSG_HDR
0231          INTEGER*4       MSG_LWD
0232          INTEGER*4       RBPCR
0233          INTEGER*4       XSTAT0
0234          INTEGER*4       XSTAT1
0235          INTEGER*4       XSTAT2
0236          INTEGER*4       XSTAT3
0237
0238          PARAMETER       REWIND = 4
0239          PARAMETER       READ = 1
0240          PARAMETER       WRITE_CHAR = 4
0241          PARAMETER       WRITE = 5
0242          PARAMETER       WRITE_SUB = 6
0243          PARAMETER       POSITION = 8
0244          PARAMETER       FORMAT = 9
0245          PARAMETER       CONTROL = 10
0246          PARAMETER       INITIALIZE = 11
0247          PARAMETER       GET_STATUS = 15
0248          PARAMETER       TIMEOUT = 96
0249          PARAMETER       XFER_ERR = 1
0250
0251          CHARACTER*16    V1CMD_HDR(0:1)
0252
0253          EQUIVALENCE     (EMB$L_DV_REGSAV(0),TSBA)
0254          EQUIVALENCE     (EMB$L_DV_REGSAV(1),TSSR)
0255          EQUIVALENCE     (EMB$L_DV_REGSAV(2),UBA_REGS(0))
0256          EQUIVALENCE     (EMB$L_DV_REGSAV(7),CMD_BUF(0))
0257          EQUIVALENCE     (EMB$L_DV_REGSAV(15),MSG_BUF(0))
0258          EQUIVALENCE     (MSG_BUF(0),MSG_HDR)
0259          EQUIVALENCE     (MSG_BUF(1),MSG_LWD)
0260          EQUIVALENCE     (MSG_BUF(2),RBPCR)
0261          EQUIVALENCE     (MSG_BUF(3),XSTAT0)
0262          EQUIVALENCE     (MSG_BUF(4),XSTAT1)
0263          EQUIVALENCE     (MSG_BUF(5),XSTAT2)
0264          EQUIVALENCE     (MSG_BUF(6),XSTAT3)
0265          EQUIVALENCE     (V2MSG_HDR(0),V1CMD_HDR(0))
0266
0267          CHARACTER*28    V1XSTAT0(0:15)
0268          DATA    V1XSTAT0(0)       /'END OF TAPE*'/
0269          DATA    V1XSTAT0(1)       /'BEGINNING OF TAPE*'/
0270          DATA    V1XSTAT0(2)       /'WRITE PROTECTED*'/
0271          DATA    V1XSTAT0(3)       /'PHASE ENCODED DRIVE*'/
0272          DATA    V1XSTAT0(4)       /'VOLUME CHECK*'/
```

```
0273          DATA     V1XSTAT0(5)      /'INTERRUPT ENABLE*'/
0274          DATA     V1XSTAT0(6)      /'DEVICE ON-LINE*'/
0275          DATA     V1XSTAT0(7)      /'TAPE MOTION ON LAST COMMAND*'/
0276          DATA     V1XSTAT0(8)      /'ILLEGAL ADDRESS*'/
0277          DATA     V1XSTAT0(9)      /'ILLEGAL COMMAND*'/
0278          DATA     V1XSTAT0(10)     /'NON-EXECUTABLE FUNCTION*'/
0279          DATA     V1XSTAT0(11)     /'WRITE LOCK ERROR*'/
0280          DATA     V1XSTAT0(12)     /'RECORD LENGTH LONG*'/
0281          DATA     V1XSTAT0(13)     /'LOGICAL END OF TAPE*'/
0282          DATA     V1XSTAT0(14)     /'RECORD LENGTH SHORT*'/
0283          DATA     V1XSTAT0(15)     /'TAPE MARK DETECTED*'/
0284
0285          CHARACTER*20    V1XSTAT1(0:7)
0286          DATA     V1XSTAT1(0)      /'MULTI-TRACK ERROR*'/
0287          DATA     V1XSTAT1(1)      /'UNCORRECTABLE ERROR*'/
0288          DATA     V1XSTAT1(2)      /'POSTAMBLE LONG*'/
0289          DATA     V1XSTAT1(3)      /'POSTAMBLE SHORT*'/
0290          DATA     V1XSTAT1(4)      /'INVALID END DATA*'/
0291          DATA     V1XSTAT1(5)      /'INVALID POSTAMBLE*'/
0292          DATA     V1XSTAT1(6)      /'SYNCH FAILURE*'/
0293          DATA     V1XSTAT1(7)      /'INVALID PREAMBLE*'/
0294
0295          CHARACTER*19    V2XSTAT1(9:13)
0296          DATA     V2XSTAT1(9)      /'SPEED CHECK*'/
0297          DATA     V2XSTAT1(10)     /'DESKEW BUFFER FAIL*'/
0298          DATA     V2XSTAT1(11)     /'TRASH IN GAP*'/
0299          DATA     V2XSTAT1(12)     /'CREASE DETECTED*'/
0300          DATA     V2XSTAT1(13)     /'CORRECTABLE ERROR*'/
0301
0302          CHARACTER*10    V3XSTAT1(15:15)
0303          DATA     V3XSTAT1(15)     /'DATA LATE*'/
0304
0305          CHARACTER*19    V1XSTAT2(10:10)
0306          DATA     V1XSTAT2(10)     /'WRITE CARD FAILURE*'/
0307
0308          CHARACTER*29    V2XSTAT2(12:15)
0309          DATA     V2XSTAT2(12)     /'CAPSTAN ACCELERATION FAILURE*'/
0310          DATA     V2XSTAT2(13)     /'SERIAL BUS PE AT DRIVE*'/
0311          DATA     V2XSTAT2(14)     /'SILO PARITY ERROR*'/
0312          DATA     V2XSTAT2(15)     /'OPERATION IN PROGRESS*'/
0313
0314          CHARACTER*25    V1XSTAT3(0:7)
0315          DATA     V1XSTAT3(0)      /'REVERSE INTO BOT*'/
0316          DATA     V1XSTAT3(1)      /'LIMIT EXCEEDED (STATIC)*'/
0317          DATA     V1XSTAT3(2)      /'NOISE RECORD*'/
0318          DATA     V1XSTAT3(3)      /'DENSITY CHECK*'/
0319          DATA     V1XSTAT3(4)      /'CAPSTAN RESPONSE FAIL*'/
0320          DATA     V1XSTAT3(5)      /'REVERSE*'/
0321          DATA     V1XSTAT3(6)      /'OPERATION INCOMPLETE*'/
0322          DATA     V1XSTAT3(7)      /'LIMIT EXCEEDED (DYNAMIC)*'/
0323
0324          CHARACTER*34    FATAL_ERROR(0:3)
0325          DATA     FATAL_ERROR(0)   /'INTERNAL MICRO DIAGNOSTIC FAILURE*'/
0326          DATA     FATAL_ERROR(1)   /'I/O SEQUENCER CROM PARITY ERROR*'/
0327          DATA     FATAL_ERROR(2)   /'MICRO PROCESSOR CROM PARITY ERROR*'/
0328          DATA     FATAL_ERROR(3)   /'AC LO*'/
0329
```

```
0330            CHARACTER*34    TERMINATION(0:7)
0331            DATA    TERMINATION(0)  /'NORMAL TERMINATION*'/
0332            DATA    TERMINATION(1)  /'ATTENTION*'/
0333            DATA    TERMINATION(2)  /'TAPE STATUS ALERT*'/
0334            DATA    TERMINATION(3)  /'FUNCTION REJECT*'/
0335            DATA    TERMINATION(4)  /'RECOVERABLE ERROR (MOTION)*'/
0336            DATA    TERMINATION(5)  /'RECOVERABLE ERROR (NO MOTION)*'/
0337            DATA    TERMINATION(6)  /'NON-RECOVERABLE ERROR (POS. LOST)*'/
0338            DATA    TERMINATION(7)  /'FATAL CONTROLLER ERROR*'/
0339
0340            CHARACTER*10    V1MSG_HDR(15:20)
0341            DATA    V1MSG_HDR(15)   /'ILLEGAL*'/
0342            DATA    V1MSG_HDR(16)   /'END*'/
0343            DATA    V1MSG_HDR(17)   /'FAIL*'/
0344            DATA    V1MSG_HDR(18)   /'ERROR*'/
0345            DATA    V1MSG_HDR(19)   /'ATTENTION*'/
0346            DATA    V1MSG_HDR(20)   /'ILLEGAL*'/
0347
0348            CHARACTER*16    V2MSG_HDR(0:1)
0349            DATA    V2MSG_HDR(0)    /'ONE WORD HEADER*'/
0350            DATA    V2MSG_HDR(1)    /'ILLEGAL*'/
0351
0352            CHARACTER*31    V3MSG_HDR(0:2)
0353            DATA    V3MSG_HDR(0)    /'ON-LINE OR OFF-LINE TRANSITION*'/
0354            DATA    V3MSG_HDR(1)    /'MICRO DIAGNOSTIC FAILURE*'/
0355            DATA    V3MSG_HDR(2)    /'ILLEGAL CLASS CODE*'/
0356
0357            CHARACTER*33    V4MSG_HDR(0:4)
0358            DATA    V4MSG_HDR(0)    /'PACKET BAD*'/
0359            DATA    V4MSG_HDR(1)    /'ILC,ILA OR NBA AFTER TAPE MOTION*'/
0360            DATA    V4MSG_HDR(2)    /'NON-EXECUTABLE FUNCTION*'/
0361            DATA    V4MSG_HDR(3)    /'MICRO DIAGNOSTIC FAILURE*'/
0362            DATA    V4MSG_HDR(4)    /'ILLEGAL CLASS CODE*'/
0363
0364            CHARACTER*33    V5MSG_HDR(15:15)
0365            DATA    V5MSG_HDR(15)   /'ACKNOWLEDGE, CPU OWNS CMD BUFFER*'/
0366
0367            CHARACTER*13    V1MSG_LWD(9:11)
0368            DATA    V1MSG_LWD(9)    /'ILLEGAL*'/
0369            DATA    V1MSG_LWD(10)   /'= 10.(BYTES)*'/
0370            DATA    V1MSG_LWD(11)   /'ILLEGAL*'/
0371
0372            CHARACTER*22    TS_COMMAND(0:11)
0373            DATA    TS_COMMAND(0)   /'INITIALIZE*'/
0374            DATA    TS_COMMAND(1)   /'READ*'/
0375            DATA    TS_COMMAND(2)   /'ILLEGAL*'/
0376            DATA    TS_COMMAND(3)   /'ILLEGAL*'/
0377            DATA    TS_COMMAND(4)   /'WRITE CHARACTERISTICS*'/
0378            DATA    TS_COMMAND(5)   /'WRITE*'/
0379            DATA    TS_COMMAND(6)   /'DIAGNOSTIC COMMAND*'/
0380            DATA    TS_COMMAND(7)   /'ILLEGAL*'/
0381            DATA    TS_COMMAND(8)   /'POSITION*'/
0382            DATA    TS_COMMAND(9)   /'FORMAT*'/
0383            DATA    TS_COMMAND(10)  /'CONTROL*'/
0384            DATA    TS_COMMAND(11)  /'ILLEGAL*'/
0385
0386            CHARACTER*17    READ_MODE(0:4)
```

```
0387              DATA      READ_MODE(0)      /'READ NEXT*'/
0388              DATA      READ_MODE(1)      /'READ PREVIOUS*'/
0389              DATA      READ_MODE(2)      /'RE-READ PREVIOUS*'/
0390              DATA      READ_MODE(3)      /'RE-READ NEXT*'/
0391              DATA      READ_MODE(4)      /'ILLEGAL MODE*'/
0392
0393              CHARACTER*17   WRITE_MODE(0:2)
0394              DATA      WRITE_MODE(0)     /'WRITE DATA*'/
0395              DATA      WRITE_MODE(1)     /'WRITE DATA RETRY*'/
0396              DATA      WRITE_MODE(2)     /'ILLEGAL MODE*'/
0397
0398              CHARACTER*24   POSITION_MODE(0:5)
0399              DATA      POSITION_MODE(0)/'SPACE RECORDS FORWARD*'/
0400              DATA      POSITION_MODE(1)/'SPACE RECORDS REVERSE*'/
0401              DATA      POSITION_MODE(2)/'SKIP TAPE MARKS FORWARD*'/
0402              DATA      POSITION_MODE(3)/'SKIP TAPE MARKS REVERSE*'/
0403              DATA      POSITION_MODE(4)/'REWIND*'/
0404              DATA      POSITION_MODE(5)/'ILLEGAL MODE*'/
0405
0406              CHARACTER*23   CONTROL_MODE(0:3)
0407              DATA      CONTROL_MODE(0) /'MESSAGE BUFFER RELEASE*'/
0408              DATA      CONTROL_MODE(1) /'REWIND AND RELOAD*'/
0409              DATA      CONTROL_MODE(2) /'CLEAN*'/
0410              DATA      CONTROL_MODE(3) /'ILLEGAL MODE*'/
0411
0412              CHARACTER*13   NORMAL_MODE(0:1)
0413              DATA      NORMAL_MODE(0)  /'NORMAL MODE*'/
0414              DATA      NORMAL_MODE(1)  /'ILLEGAL MODE*'/
0415
0416              CHARACTER*22   FORMAT_MODE(0:3)
0417              DATA      FORMAT_MODE(0)  /'WRITE TAPE MARK*'/
0418              DATA      FORMAT_MODE(1)  /'ERASE*'/
0419              DATA      FORMAT_MODE(2)  /'WRITE TAPE MARK ENTRY*'/
0420              DATA      FORMAT_MODE(3)  /'ILLEGAL MODE*'/
0421
0422              CHARACTER*17   V2CMD_HDR(7:7)
0423              DATA      V2CMD_HDR(7)    /'INTERRUPT ENABLE*'/
0424
0425              CHARACTER*34   V3CMD_HDR(12:15)
0426              DATA      V3CMD_HDR(12)   /'SWAP BYTES*'/
0427              DATA      V3CMD_HDR(13)   /'OPPOSITE*'/
0428              DATA      V3CMD_HDR(14)   /'CLEAR VOLUME CHECK*'/
0429              DATA      V3CMD_HDR(15)   /'ACKNOWLEDGE, TS11 OWNS MSG BUFFER*'/
0430
0431              CHARACTER*17   V1TSSR(6:7)
0432              DATA      V1TSSR(6)       /'OFF LINE*'/
0433              DATA      V1TSSR(7)       /'SUB-SYSTEM READY*'/
0434
0435              CHARACTER*30   V2TSSR(10:15)
0436              DATA      V2TSSR(10)      /'NEED BUFFER ADDRESS*'/
0437              DATA      V2TSSR(11)      /'NON-EXISTENT MEMORY*'/
0438              DATA      V2TSSR(12)      /'REGISTER MODIFICATION REFUSED*'/
0439              DATA      V2TSSR(13)      /'SERIAL BUS PARITY ERROR*'/
0440              DATA      V2TSSR(14)      /'UNIBUS PARITY ERROR*'/
0441              DATA      V2TSSR(15)      /'SPECIAL CONDITION*'/
0442
0443              CHARACTER*2    EX_ADDR(8:9)
```

TSTAPE

F 9
16-Sep-1984 00:15:51
5-Sep-1984 14:23:51
VAX-11 FORTRAN V3.4-56
DISK$VMSMASTER:[ERF.SRC]TSTAPE.FOR;1
Page 6

```
0444              DATA    EX_ADDR(8)      /'16'/
0445              DATA    EX_ADDR(9)      /'17'/
0446
0447              CHARACTER*7    V1MODE
0448              DATA    V1MODE          /'MODE = '/
0449
0450
0451              CALL FRCTOF (LUN)
0452
0453              call dhead1 (lun,'UBA TS11')
0454
0455              CALL LINCHK (LUN,2)
0456
0457              WRITE(LUN,10) TSBA
0458      10      FORMAT(/' ',T8,'TSBA',T24,Z8.4)
0459
0460              CALL LINCHK (LUN,1)
0461
0462              WRITE(LUN,15) TSSR
0463      15      FORMAT(' ',T8,'TSSR',T24,Z8.4)
0464
0465              IF (JIAND(TSSR,'800E'X) .EQ. '800E'X) THEN
0466
0467              FIELD = LIB$EXTZV(4,2,TSSR)
0468
0469              CALL LINCHK (LUN,2)
0470
0471              WRITE(LUN,20) FATAL_ERROR(FIELD)
0472      20      FORMAT(' ',T40,A<COMPRESSC (FATAL_ERROR(FIELD))>,/,
0473            1 T40,'FATAL CONTROLLER ERROR')
0474              ELSE
0475
0476              FIELD = LIB$EXTZV(1,3,TSSR)
0477
0478              CALL LINCHK (LUN,1)
0479
0480              WRITE(LUN,25) TERMINATION(FIELD)
0481      25      FORMAT(' ',T40,A<COMPRESSC (TERMINATION(FIELD))>)
0482              ENDIF
0483
0484              CALL OUTPUT (LUN,TSSR,V1TSSR,6,7,7,'0')
0485
0486              DO 28,I = 8,9
0487
0488              IF (JIAND(TSSR,2**I) .NE. 0) THEN
0489
0490              CALL LINCHK (LUN,1)
0491
0492              WRITE(LUN,27) EX_ADDR(I)
0493      27      FORMAT(' ',T40,'EXTENDED BUS ADDRESS BIT ',A2,'.')
0494              ENDIF
0495
0496      28      CONTINUE
0497
0498              CALL OUTPUT (LUN,TSSR,V2TSSR,10,10,15,'0')
0499
0500              CALL LINCHK (LUN,3)
```

```
0501
0502              WRITE(LUN,30)
0503     30       FORMAT(/' ','COMMAND BUFFER',/)
0504
0505              CALL LINCHK (LUN,1)
0506
0507              WRITE(LUN,35) CMD_BUF(0)
0508     35       FORMAT(' ',T8,'CMD HDR',T24,Z8.4)
0509
0510              TS_FUNCTION = LIB$EXTZV(0,5,CMD_BUF(0))
0511
0512              CALL LINCHK (LUN,1)
0513
0514              WRITE(LUN,40) TS_COMMAND(MIN(11,TS_FUNCTION))
0515     40       FORMAT(' ',T40,'FUNCTION = ',
0516            1 A<COMPRESSC (TS_COMMAND(MIN(11,TS_FUNCTION)))>)
0517
0518              FIELD = LIB$EXTZV(5,2,CMD_BUF(0))
0519
0520              CALL LINCHK (LUN,1)
0521
0522              WRITE(LUN,45) V1CMD_HDR(MIN(1,FIELD))
0523     45       FORMAT(' ',T40,A<COMPRESSC (V1CMD_HDR(MIN(1,FIELD)))>)
0524
0525              CALL OUTPUT (LUN,CMD_BUF(0),V2CMD_HDR,7,7,7,'0')
0526
0527              IF (TS_FUNCTION .EQ. INITIALIZE
0528            1 .OR.
0529            2 TS_FUNCTION .EQ. CONTROL
0530            3 .OR.
0531            4 TS_FUNCTION .EQ. FORMAT
0532            5 .OR.
0533            6 TS_FUNCTION .EQ. POSITION
0534            7 .OR.
0535            8 TS_FUNCTION .EQ. WRITE_SUB
0536            9 .OR.
0537            1 TS_FUNCTION .EQ. WRITE
0538            2 .OR.
0539            3 TS_FUNCTION .EQ. WRITE_CHAR
0540            4 .OR.
0541            5 TS_FUNCTION .EQ. READ
0542            6 .OR.
0543            7 TS_FUNCTION .EQ. GET_STATUS) THEN
0544
0545              CALL LINCHK (LUN,1)
0546
0547              MODE = LIB$EXTZV(8,4,CMD_BUF(0))
0548
0549              IF (TS_FUNCTION .EQ. INITIALIZE
0550            1 .OR.
0551            2 TS_FUNCTION .EQ. WRITE_SUB
0552            3 .OR.
0553            4 TS_FUNCTION .EQ. WRITE_CHAR
0554            5 .OR.
0555            6 TS_FUNCTION .EQ. GET_STATUS) THEN
0556
0557              WRITE(LUN,50) V1MODE,NORMAL_MODE(MIN(1,MODE))
```

```
0558     50        FORMAT(' ',T40,A7,A<COMPRESSC (NORMAL_MODE(MIN(1,MODE)))>)
0559
0560               ELSE IF (TS_FUNCTION .EQ. CONTROL) THEN
0561
0562               WRITE(LUN,55) V1MODE,CONTROL_MODE(MIN(4,MODE))
0563     55        FORMAT(' ',T40,A7,A<COMPRESSC (CONTROL_MODE(MIN(4,MODE)))>)
0564
0565               ELSE IF (TS_FUNCTION .EQ. FORMAT) THEN
0566
0567               WRITE(LUN,60) V1MODE,FORMAT_MODE(MIN(3,MODE))
0568     60        FORMAT(' ',T40,A7,A<COMPRESSC (FORMAT_MODE(MIN(3,MODE)))>)
0569
0570               ELSE IF (TS_FUNCTION .EQ. POSITION) THEN
0571
0572               WRITE(LUN,65) V1MODE,POSITION_MODE(MIN(5,MODE))
0573     65        FORMAT(' ',T40,A7,A<COMPRESSC (POSITION_MODE(MIN(5,MODE)))>)
0574
0575               ELSE IF (TS_FUNCTION .EQ. WRITE) THEN
0576
0577               WRITE(LUN,70) V1MODE,WRITE_MODE(MIN(3,MODE))
0578     70        FORMAT(' ',T40,A7,A<COMPRESSC (WRITE_MODE(MIN(3,MODE)))>)
0579
0580               ELSE IF (TS_FUNCTION .EQ. READ) THEN
0581
0582               WRITE(LUN,75) V1MODE,READ_MODE(MIN(4,MODE))
0583     75        FORMAT(' ',T40,A7,A<COMPRESSC (READ_MODE(MIN(4,MODE)))>)
0584               ENDIF
0585               ENDIF
0586
0587               CALL OUTPUT (LUN,CMD_BUF(0),V3CMD_HDR,12,12,15,'0')
0588
0589               IF (TS_FUNCTION .EQ. WRITE
0590              1 .OR.
0591              2 TS_FUNCTION .EQ. WRITE_CHAR
0592              3 .OR.
0593              4 TS_FUNCTION .EQ. READ) THEN
0594
0595               CALL LINCHK (LUN,1)
0596
0597               WRITE(LUN,80) CMD_BUF(1)
0598     80        FORMAT(' ',T8,'LO ADDR BITS',T24,Z8.4)
0599
0600               CALL LINCHK (LUN,1)
0601
0602               WRITE(LUN,85) CMD_BUF(2)
0603     85        FORMAT(' ',T8,'HI ADDR BITS',T24,Z8.4)
0604
0605               CALL CALC_MAP (LUN,0,CMD_BUF(2),CMD_BUF(1))
0606
0607               CALL LINCHK (LUN,1)
0608
0609               WRITE(LUN,90) CMD_BUF(3)
0610     90        FORMAT(' ',T8,'SIZE',T24,Z8.4)
0611
0612               CALL LINCHK (LUN,1)
0613
0614               WRITE(LUN,95) CMD_BUF(3)
```

```
0615    95        FORMAT(' ',T40,'BUFFER SIZE = ',I<COMPRESS4 (CMD_BUF(3))>,'. BYTES')
0616
0617              ELSE IF (TS_FUNCTION .EQ. POSITION) THEN
0618
0619              CALL LINCHK (LUN,1)
0620
0621              WRITE(LUN,98) CMD_BUF(1)
0622    98        FORMAT(' ',T8,'SKIP/SPACE COUNT',T24,Z8.4)
0623
0624              CALL LINCHK (LUN,1)
0625
0626              WRITE(LUN,100) CMD_BUF(1)
0627    100       FORMAT(' ',T40,'SKIP/SPACE ',I<COMPRESS4 (CMD_BUF(1))>
0628            1 ,'. TAPE MARKS/REC')
0629              ENDIF
0630
0631              IF (EMB$W_HD_ENTRY .NE. TIMEOUT) THEN
0632
0633              CALL LINCHK (LUN,3)
0634
0635              IF (JIAND(TSSR,'400'X) .NE. 0) THEN
0636
0637              WRITE(LUN,105)
0638    105       FORMAT(' ','MESSAGE BUFFER NOT VALID',/)
0639              ELSE
0640
0641              WRITE(LUN,110)
0642    110       FORMAT(/' ','MESSAGE BUFFER',/)
0643
0644              CALL LINCHK (LUN,1)
0645
0646              WRITE(LUN,115) MSG_HDR
0647    115       FORMAT(' ',T8,'MSG HDR',T24,Z8.4)
0648
0649              CALL LINCHK (LUN,1)
0650
0651              FIELD = LIB$EXTZV(0,5,MSG_HDR)
0652
0653              WRITE(LUN,120) V1MSG_HDR(MAX(15,MIN(20,FIELD)))
0654    120       FORMAT(' ',T40,'MESSAGE TYPE = ',
0655            1 A<COMPRESSC (V1MSG_HDR(MAX(15,MIN(20,FIELD))))>)
0656
0657              CALL LINCHK (LUN,1)
0658
0659              FIELD = LIB$EXTZV(5,3,MSG_HDR)
0660
0661              WRITE(LUN,125) V2MSG_HDR(MIN(1,FIELD))
0662    125       FORMAT(' ',T40,A<COMPRESSC (V2MSG_HDR(MIN(1,FIELD)))>,
0663            1 ' PACKET FORMAT')
0664
0665              IF (JIAND(MSG_HDR,'11'X) .EQ. '11'X) THEN
0666
0667              FIELD = LIB$EXTZV(8,4,MSG_HDR)
0668
0669              CALL LINCHK (LUN,1)
0670
0671              IF (JIAND(MSG_HDR,'2'X) .EQ. '2'X) THEN
```

```
0672
0673            WRITE(LUN,130) V3MSG_HDR(MIN(2,FIELD))
0674    130     FORMAT(' ',T40,A<COMPRESSC (V3MSG_HDR(MIN(2,FIELD)))>)
0675            ELSE
0676
0677            WRITE(LUN,135) V4MSG_HDR(MIN(4,FIELD))
0678    135     FORMAT(' ',T40,A<COMPRESSC (V4MSG_HDR(MIN(4,FIELD)))>)
0679            ENDIF
0680            ENDIF
0681
0682            CALL OUTPUT (LUN,MSG_HDR,V5MSG_HDR,15,15,15,'0')
0683
0684            CALL LINCHK (LUN,1)
0685
0686            WRITE(LUN,140) MSG_LWD
0687    140     FORMAT(' ',T8,'MSG-LEN',T24,Z8.4)
0688
0689            CALL LINCHK (LUN,1)
0690
0691            FIELD = LIB$EXTZV(0,16,MSG_LWD)
0692
0693            WRITE(LUN,145) V1MSG_LWD(MAX(9,MIN(11,FIELD)))
0694    145     FORMAT(' ',T40,'MESSAGE LENGTH '
0695          1 A<COMPRESSC (V1MSG_LWD(MAX(9,MIN(11,FIELD))))>)
0696
0697            CALL LINCHK (LUN,1)
0698
0699            WRITE(LUN,150) RBPCR
0700    150     FORMAT(' ',T8,'RBPCR',T24,Z8.4)
0701
0702            FIELD = LIB$EXTZV(0,16,RBPCR)
0703
0704            IF (TS_FUNCTION .EQ. READ) THEN
0705
0706            CALL LINCHK (LUN,1)
0707
0708            WRITE(LUN,152) FIELD
0709    152     FORMAT(' ',T40,'RES. BYTES OF XFER = ',I<COMPRESS4 (FIELD)>,'.')
0710
0711            ELSE IF (TS_FUNCTION .EQ. POSITION
0712          1 .AND.
0713          2 MODE .NE. REWIND) THEN
0714
0715            CALL LINCHK (LUN,1)
0716
0717            WRITE(LUN,154) FIELD
0718    154     FORMAT(' ',T40,'RES. REC/MARKS OF SKIP = ',I<COMPRESS4 (FIELD)>,'.')
0719            ENDIF
0720
0721            CALL LINCHK (LUN,1)
0722
0723            WRITE(LUN,155) XSTAT0
0724    155     FORMAT(' ',T8,'XSTAT0',T24,Z8.4)
0725
0726            CALL OUTPUT (LUN,XSTAT0,V1XSTAT0,0,0,15,'0')
0727
0728            CALL LINCHK (LUN,1)
```

```
0729
0730            WRITE(LUN,160) XSTAT1
0731     160    FORMAT(' ',T8,'XSTAT1',T24,Z8.4)
0732
0733            CALL OUTPUT (LUN,XSTAT1,V1XSTAT1,0,0,7,'0')
0734
0735            CALL OUTPUT (LUN,XSTAT1,V2XSTAT1,9,9,13,'0')
0736
0737            CALL OUTPUT (LUN,XSTAT1,V3XSTAT1,15,15,15,'0')
0738
0739            CALL LINCHK (LUN,1)
0740
0741            WRITE(LUN,165) XSTAT2
0742     165    FORMAT(' ',T8,'XSTAT2',T24,Z8.4)
0743
0744            DO 178,I = 0,8
0745
0746            IF (JIAND(XSTAT2,2**I) .NE. 0) THEN
0747
0748            CALL LINCHK (LUN,1)
0749
0750            IF (I .NE. 8) THEN
0751
0752            WRITE(LUN,170) I
0753     170    FORMAT(' ',T40,'DEAD TRACK CHANNEL ',I1,'.')
0754            ELSE
0755
0756            WRITE(LUN,175)
0757     175    FORMAT(' ',T40,'DEAD TRACK PARITY CHANNEL')
0758            ENDIF
0759            ENDIF
0760
0761     178    CONTINUE
0762
0763            CALL OUTPUT (LUN,XSTAT2,V1XSTAT2,10,10,10,'0')
0764
0765            CALL OUTPUT (LUN,XSTAT2,V2XSTAT2,12,12,15,'0')
0766
0767            CALL LINCHK (LUN,1)
0768
0769            WRITE(LUN,180) XSTAT3
0770     180    FORMAT(' ',T8,'XSTAT3',T24,Z8.4)
0771
0772            CALL OUTPUT (LUN,XSTAT3,V1XSTAT3,0,0,7,'0')
0773
0774            FIELD = LIB$EXTZV(8,8,XSTAT3)
0775
0776            IF (FIELD .NE. 0) THEN
0777
0778            CALL LINCHK (LUN,1)
0779
0780            WRITE(LUN,185) FIELD
0781     185    FORMAT(' ',T40,'MICRO DIAGNOSTIC ERROR CODE ',O3.3)
0782            ENDIF
0783            ENDIF
0784
0785
```

```
0786              if (
0787            1 ((ts_function .eq. read
0788            1 .or.
0789            1 ts_function .eq. write
0790            1 .or.
0791            1 ts_function .eq. write_char
0792            1 .or.
0793            1 ts_function .eq. write_sub)
0794            1 .and.
0795            1 emb$w_hd_entry .ne. 96)
0796            1 .or.
0797            1 emb$w_hd_entry .ne. 98
0798            1 ) then
0799
0800      C
0801      C       IF THE TS11 FUNCTION IS A READ REVERSE THEN
0802      C       THE PREVIOUS MAP IS THE FINAL MAP PLUS ONE.
0803      C
0804
0805              if (uba_regs(0) .ne. 0) then
0806
0807              call uba_datapath (lun,uba_regs(0),uba_regs(1))
0808              endif
0809
0810              call calc_map2 (0,cmd_buf(2),cmd_buf(1),field)
0811
0812              call uba_mapping (lun,field,uba_regs(2))
0813
0814              if (lib$extzv (16,16,emb$l_dv_iosb1) .gt. 512) then
0815
0816              IF (TS_FUNCTION .EQ. READ
0817            1 .AND.
0818            2 (MODE .EQ. 1
0819            3 .OR.
0820            4 MODE .EQ. 2)) THEN
0821
0822              call uba_mapping (lun,(field+2),uba_regs(4))
0823              ELSE
0824
0825              call uba_mapping (lun,(field+1),uba_regs(3))
0826              ENDIF
0827              endif
0828              ENDIF
0829              endif
0830
0831              call linchk (lun,1)
0832
0833              write(lun,190)
0834      190     format(' ',:)
0835
0836              if (emb$w_hd_entry .ne. 98) then
0837
0838              call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
0839
0840              call ucb$b_ertmax (lun,emb$b_dv_ertmax)
0841              endif
0842
```

```
0843            call orb$l_owner (lun,emb$l_dv_ownuic)
0844
0845            call ucb$l_char (lun,emb$l_dv_char)
0846
0847            call ucb$w_sts (lun,emb$w_dv_sts)
0848
0849            call ucb$l_opcnt (lun,emb$l_dv_opcnt)
0850
0851            call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0852
0853            if (emb$w_hd_entry .ne. 98) then
0854
0855            call linchk (lun,1)
0856
0857            write(lun,190)
0858
0859            call tstape_qio (lun,emb$w_dv_func)
0860
0861            call irp$w_bcnt (lun,emb$w_dv_bcnt)
0862
0863            call irp$w_boff (lun,emb$w_dv_boff)
0864
0865            call irp$l_pid (lun,emb$l_dv_rqpid)
0866
0867            call irp$q_iosb (lun,emb$l_dv_iosb1)
0868            endif
0869
0870            RETURN
0871            END
```

PROGRAM SECTIONS

```
   Name                             Bytes    Attributes

 0 $CODE                            4242     PIC CON REL LCL    SHR    EXE    RD NOWRT LONG
 1 $PDATA                           1020     PIC CON REL LCL    SHR  NOEXE    RD NOWRT LONG
 2 $LOCAL                           4504     PIC CON REL LCL  NOSHR  NOEXE    RD   WRT LONG
 3 EMB                               512     PIC OVR REL GBL    SHR  NOEXE    RD   WRT LONG

   Total Space Allocated          10278
```

ENTRY POINTS

```
   Address   Type   Name

 0-00000000         TSTAPE
```

VARIABLES

```
   Address   Type   Name                         Address   Type   Name

 3-0000001C  L*1    EMB$B_DV_CLASS              3-00000010  L*1    EMB$B_DV_ERTCNT
 3-00000011  L*1    EMB$B_DV_ERTMAX            3-0000003E  L*1    EMB$B_DV_NAMLNG
 3-0000003A  L*1    EMB$B_DV_SLAVE             3-0000001D  L*1    EMB$B_DV_TYPE
 3-00000036  I*4    EMB$L_DV_CHAR              3-00000012  I*4    EMB$L_DV_IOSB1
 3-00000016  I*4    EMB$L_DV_IOSB2             3-00000026  I*4    EMB$L_DV_MEDIA
 3-0000004E  I*4    EMB$L_DV_NUMREG            3-0000002E  I*4    EMB$L_DV_OPCNT
 3-00000032  I*4    EMB$L_DV_OWNUIC            3-0000001E  I*4    EMB$L_DV_RQPID
 3-00000000  I*4    EMB$L_HD_SID               3-0000003F  CHAR   EMB$T_DV_NAME
 3-00000024  I*2    EMB$W_DV_BCNT              3-00000022  I*2    EMB$W_DV_BOFF
 3-0000002C  I*2    EMB$W_DV_ERRCNT            3-0000003C  I*2    EMB$W_DV_FUNC
 3-0000001A  I*2    EMB$W_DV_STS               3-0000002A  I*2    EMB$W_DV_UNIT
 3-00000004  I*2    EMB$W_HD_ENTRY             3-0000000E  I*2    EMB$W_HD_ERRSEQ
 2-00000BC4  I*4    FIELD                      2-00000BCC  I*4    I
 AP-0000004B L*1    LUN                        2-00000BC0  I*4    MODE
 3-0000008E  I*4    MSG_HDR                    3-00000092  I*4    MSG_LWD
 3-00000096  I*4    RBPCR                      3-00000052  I*4    TSBA
 3-00000056  I*4    TSSR                       2-00000BC8  I*4    TS_FUNCTION
 2-00000BB7  CHAR   V1MODE                     3-0000009A  I*4    XSTAT0
 3-0000009E  I*4    XSTAT1                     3-000000A2  I*4    XSTAT2
 3-000000A6  I*4    XSTAT3
```

ARRAYS

```
   Address   Type   Name                         Bytes   Dimensions

 3-0000006E  I*4    CMD_BUF                         32    (0:7)
 2-00000976  CHAR   CONTROL_MODE                    92    (0:3)
 3-00000000  L*1    EMB                            512    (0:511)
 3-00000052  I*4    EMB$L_DV_REGSAV                420    (0:104)
 3-00000006  I*4    EMB$Q_HD_TIME                    8    (2)
 2-00000BB3  CHAR   EX_ADDR                          4    (8:9)
 2-00000438  CHAR   FATAL_ERROR                    136    (0:3)
 2-000009EC  CHAR   FORMAT_MODE                     88    (0:3)
```

```
3-0000008E  I*4  MSG_BUF                              28   (0:6)
2-000009D2  CHAR NORMAL_MODE                          26   (0:1)
2-000008E6  CHAR POSITION_MODE                       144   (0:5)
2-0000085E  CHAR READ_MODE                            85   (0:4)
2-000004C0  CHAR TERMINATION                         272   (0:7)
2-00000756  CHAR TS_COMMAND                          264   (0:11)
3-0000005A  I*4  UBA_REGS                              20   (0:4)
2-00000000  CHAR V1CMD_HDR                            32   (0:1)
2-000005D0  CHAR V1MSG_HDR                            60   (15:20)
2-0000072F  CHAR V1MSG_LWD                            39   (9:11)
2-00000ADD  CHAR V1TSSR                               34   (6:7)
2-00000020  CHAR V1XSTAT0                            448   (0:15)
2-000001E0  CHAR V1XSTAT1                            160   (0:7)
2-000002E9  CHAR V1XSTAT2                             19   (10:10)
2-00000370  CHAR V1XSTAT3                            200   (0:7)
2-00000A44  CHAR V2CMD_HDR                            17   (7:7)
2-00000000  CHAR V2MSG_HDR                            32   (0:1)
2-00000AFF  CHAR V2TSSR                              180   (10:15)
2-00000280  CHAR V2XSTAT1                             95   (9:13)
2-000002FC  CHAR V2XSTAT2                            116   (12:15)
2-00000A55  CHAR V3CMD_HDR                           136   (12:15)
2-0000060C  CHAR V3MSG_HDR                            93   (0:2)
2-000002DF  CHAR V3XSTAT1                             10   (15:15)
2-00000669  CHAR V4MSG_HDR                           165   (0:4)
2-0000070E  CHAR V5MSG_HDR                            33   (15:15)
2-000008B3  CHAR WRITE_MODE                           51   (0:2)
```

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000047	10'	1-00000059	15'	1-0000006A	20'	1-00000091	25'	1-0000009D	27'	**	28'
1-000000C3	30'	1-000000D9	35'	1-000000ED	40'	1-00000106	45'	1-00000112	50'	1-00000120	55'
1-0000012E	60'	1-0000013C	65'	1-0000014A	70'	1-00000158	75'	1-00000166	80'	1-0000017F	85'
1-00000198	90'	1-000001A9	95'	1-000001CE	98'	1-000001EB	100'	1-00000216	105'	1-00000235	110'
1-0000024B	115'	1-0000025F	120'	1-0000027C	125'	1-00000298	130'	1-000002A4	135'	1-000002B0	140'
1-000002C4	145'	1-000002E1	150'	1-000002F3	152'	1-00000319	154'	1-00000343	155'	1-00000356	160'
1-00000369	165'	1-0000037C	170'	1-0000039C	175'	**	178	1-000003BD	180'	1-000003D0	185'
1-000003F7	190'										

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
	CALC_MAP		CALC_MAP2	I*4	COMPRESS4	I*4	COMPRESSC		DHEAD1		FRCTOF
	IRP$C_PID		IRP$Q_IOSB		IRP$W_BCNT		IRP$W_BOFF	I*4	LIB$EXTZV		LINCHK
	ORB$L_OWNER		OUTPUT		TSTAPE_QIO		UBA_DATAPATH		UBA_MAPPING		UCB$B_ERTCNT
	UCB$B_ERTMAX		UCB$L_CHAR		UCB$L_OPCNT		UCB$W_ERRCNT		UCB$W_STS		

```
0001
0002
0003            Subroutine TSTAPE_QIO (lun,emb$w_dv_func)
0004
0005
0006            include 'src$:qiocommon.for /nolist'
0270
0271
0272            byte            lun
0273
0274            integer*2       emb$w_dv_func
0275
0276            integer*4       qiocode(0:1,0:63)
0277
0278
0279            if (qiocode(0,0) .eq. 0) then
0280
0281            qiocode(1,00) = %loc(io$_nop)
0282
0283            qiocode(1,01) = %loc(io$_unload)
0284
0285            qiocode(1,02) = %loc(io$_spacefile)
0286
0287            qiocode(1,03) = %loc(io$_recal)
0288
0289            qiocode(1,06) = %loc(io$_erasetape)
0290
0291            qiocode(1,08) = %loc(io$_packack)
0292
0293            qiocode(1,09) = %loc(io$_spacerecord)
0294
0295            qiocode(1,10) = %loc(io$_writecheck)
0296
0297            qiocode(1,11) = %loc(io$_writepblk)
0298
0299            qiocode(1,12) = %loc(io$_readpblk)
0300
0301            qiocode(1,22) = %loc(io$_rereadn)
0302
0303            qiocode(1,23) = %loc(io$_rereadp)
0304
0305            qiocode(1,24) = %loc(io$_writeret)
0306
0307            qiocode(1,25) = %loc(io$_readpreset)
0308
0309            qiocode(1,26) = %loc(io$_setchar)
0310
0311            qiocode(1,27) = %loc(io$_sensechar)
0312
0313            qiocode(1,28) = %loc(io$_writemark)
0314
0315            qiocode(1,29) = %loc(io$_wrttmkr)
0316
0317            qiocode(1,30) = %loc(io$_clean)
0318
0319            qiocode(1,32) = %loc(io$_writelblk)
0320
```

```
0321            qiocode(1,33) = %loc(io$_readlblk)
0322
0323            qiocode(1,34) = %loc(io$_rewindoff)
0324
0325            qiocode(1,35) = %loc(io$_setmode)
0326
0327            qiocode(1,36) = %loc(io$_rewind)
0328
0329            qiocode(1,37) = %loc(io$_skipfile)
0330
0331            qiocode(1,38) = %loc(io$_skiprecord)
0332
0333            qiocode(1,39) = %loc(io$_sensemode)
0334
0335            qiocode(1,40) = %loc(io$_writeof)
0336
0337            qiocode(1,48) = %loc(io$_writevblk)
0338
0339            qiocode(1,49) = %loc(io$_readvblk)
0340
0341            qiocode(1,50) = %loc(io$_access)
0342
0343            qiocode(1,51) = %loc(io$_create)
0344
0345            qiocode(1,52) = %loc(io$_deaccess)
0346
0347            qiocode(1,53) = %loc(io$_delete)
0348
0349            qiocode(1,54) = %loc(io$_modify)
0350
0351            qiocode(1,56) = %loc(io$_acpcontrol)
0352
0353            qiocode(1,57) = %loc(io$_mount)
0354
0355            do 10,i = 0,63
0356
0357            qiocode(0,i) = 33
0358
0359            if (qiocode(1,i) .eq. 0) then
0360
0361            qiocode(1,i) = %loc(qio_string)
0362            endif
0363
0364    10      continue
0365            endif
0366
0367            call irp$w_func (lun,emb$w_dv_func,
0368          1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0369
0370            return
0371
0372            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes								
0	$CODE	346	PIC	CON REL LCL		SHR	EXE	RD	NOWRT	LONG
1	$PDATA	8	PIC	CON REL LCL		SHR	NOEXE	RD	NOWRT	LONG
2	$LOCAL	548	PIC	CON REL LCL	NOSHR	NOEXE	RD	WRT	LONG
3	QIOCOMMON	1247	PIC	OVR REL GBL		SHR	NOEXE	RD	WRT	LONG

Total Space Allocated   2149

ENTRY POINTS

Address	Type	Name
0-00000000		TSTAPE_QIO

VARIABLES

Address	Type	Name		Address	Type	Name
AP-0000008a	I*2	EMBSW_DV_FUNC		2-00000200	I*4	I
3-00000442	CHAR	IO$_ABORT		3-0000034D	CHAR	IO$_ACCESS
3-000003C2	CHAR	IO$_ACPCONTROL		3-000004B3	CHAR	IO$_AVAILABLE
3-00000297	CHAR	IO$_CLEAN		3-00000369	CHAR	IO$_CREATE
3-00000385	CHAR	IO$_DEACCESS		3-00000393	CHAR	IO$_DELETE
3-0000026D	CHAR	IO$_DIAGNOSE		3-00000065	CHAR	IO$_DRVCLR
3-000004CB	CHAR	IO$_DSE		3-000000A9	CHAR	IO$_ERASETAPE
3-00000276	CHAR	IO$_FORMAT		3-00000071	CHAR	IO$_INITIALIZE
3-00000014	CHAR	IO$_LOADMCODE		3-000003A1	CHAR	IO$_MODIFY
3-000003E2	CHAR	IO$_MOUNT		3-00000000	CHAR	IO$_NOP
3-0000009D	CHAR	IO$_OFFSET		3-000000EB	CHAR	IO$_PACKACK
3-000000E0	CHAR	IO$_QSTOP		3-000003EF	CHAR	IO$_RDSTATS
3-00000421	CHAR	IO$_READCSR		3-00000169	CHAR	IO$_READHEAD
3-000002B6	CHAR	IO$_READLBLK		3-0000013F	CHAR	IO$_READPBLK
3-00000200	CHAR	IO$_READPRESET		3-00000195	CHAR	IO$_READTRACKD
3-0000033A	CHAR	IO$_READVBLK		3-0000045A	CHAR	IO$_READWTHBUF
3-00000484	CHAR	IO$_READWTHXBUF		3-0000004D	CHAR	IO$_RECAL
3-0000007C	CHAR	IO$_RELEASE		3-000001AB	CHAR	IO$_REREADN
3-000001B8	CHAR	IO$_REREADP		3-000000CA	CHAR	IO$_RETCENTER
3-000002E6	CHAR	IO$_REWIND		3-000002C9	CHAR	IO$_REWINDOFF
3-000000FC	CHAR	IO$_SEARCH		3-00000024	CHAR	IO$_SEEK
3-00000231	CHAR	IO$_SENSECHAR		3-00000309	CHAR	IO$_SENSEMODE
3-0000021D	CHAR	IO$_SETCHAR		3-000003B8	CHAR	IO$_SETCLOCK
3-00000088	CHAR	IO$_SETCLOCKP		3-000002DD	CHAR	IO$_SETMODE
3-000002ED	CHAR	IO$_SKIPFILE		3-000002FA	CHAR	IO$_SKIPRECORD
3-00000029	CHAR	IO$_SPACEFILE		3-0000010E	CHAR	IO$_SPACERECORD
3-000003D7	CHAR	IO$_STARTDATA		3-000000B4	CHAR	IO$_STARTDATAP
3-00000037	CHAR	IO$_STARTMPROC		3-0000020F	CHAR	IO$_STARTSPNDL
3-00000059	CHAR	IO$_STOP		3-0000000D	CHAR	IO$_UNLOAD
3-0000046B	CHAR	IO$_WRITEBUFNCRC		3-0000011E	CHAR	IO$_WRITECHECK
3-000001E4	CHAR	IO$_WRITECHECKH		3-000003FF	CHAR	IO$_WRITECSR
3-00000153	CHAR	IO$_WRITEHEAD		3-000002A2	CHAR	IO$_WRITELBLK
3-00000247	CHAR	IO$_WRITEMARK		3-00000314	CHAR	IO$_WRITEOF
3-0000012A	CHAR	IO$_WRITEPBLK		3-000001C9	CHAR	IO$_WRITERET

```
3-0000017E  CHAR IOS_WRITETRACKD          3-00000326  CHAR IOS_WRITEVBLK
3-00000448  CHAR IOS_WRITEWTHBUF          3-00000257  CHAR IOS_WRTTMKR
AP-0000004a L*1  LUN                      3-000004A1  CHAR QIO_STRING
```

ARRAYS

```
   Address  Type  Name              Bytes  Dimensions

   2-00000000  I*4  QIOCODE          512   (0:1, 0:63)
```

LABELS

```
   Address  Label

      **      10
```

FUNCTIONS AND SUBROUTINES REFERENCED

```
   Type  Name            Type  Name

         IRP$W_FUNC      I*4  LIB$EXTZV
```

COMMAND QUALIFIERS

```
  FORTRAN /LIS=LIS$:TSTAPE/OBJ=OBJ$:TSTAPE MSRC$:TSTAPE

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE /NOMACHINE_CODE /CONTINUATIONS=19
```

COMPILATION STATISTICS

```
  Run Time:        14.63 seconds
  Elapsed Time:    39.25 seconds
  Page Faults:     289
  Dynamic Memory:  266 pages
```

TIMCMP
LIS

TUTAPE
LIS

SBI
LIS

STSEVENT
LIS

TRNS.BITS
LIS

UBAERR
LIS

TIMRB
LIS

UBA
LIS

TU81SENSE
LIS

UNDEFINED
LIS

SYSPWRFL
LIS

SUMMARY
LIS

UNKN DISP
LIS

SHRVECTOR
LIS

SYSTARTUP
LIS

UBAINT
LIS

TOF
LIS

UNKNOWN
LIS

TSTAPE
LIS